

PYTHON 13 STYCZNIA

---

# PRAKTYCZNE ZASTOSOWANIA NA PRZYKŁADACH

WPROWADZENIE BYŁO, DZISIAJ WIĘCEJ PRZYKŁADÓW

---

## PRZYPOMNIENIE O INSTALACJI

- ▶ `pip install pysimplegui`
- ▶ `conda install pysimplegui`
- ▶ W Mac OS może być problem z pierwszym.

WPROWADZENIE BYŁO, DZISIAJ WIĘCEJ PRZYKŁADÓW

---

## PODSTAWOWE ELEMENTY INTERFEJSU PYSIMPLEGUI:

- ▶ Layout: układ elementów GUI (np. przyciski, pola tekstowe).
- ▶ Window: okno, które wyświetla elementy.
- ▶ Event Loop: pętla obsługująca interakcje użytkownika.

WPROWADZENIE BYŁO, DZISIAJ WIĘCEJ PRZYKŁADÓW

---

## PODSTAWOWE ELEMENTY INTERFEJSU PYSIMPLEGUI:

- ▶ Layout: układ elementów GUI (np. przyciski, pola tekstowe).
- ▶ Window: okno, które wyświetla elementy.
- ▶ Event Loop: pętla obsługująca interakcje użytkownika.

## PRZYKŁAD 1 - KALKULATOR

```
import PySimpleGUI as sg

# Definicja układu interfejsu (layout)
# Layout to lista list - każda wewnętrzna lista odpowiada jednemu wierszowi GUI.
layout = [
    [sg.Text("Podaj dwie liczby:")], # Etykieta tekstowa
    [sg.Input(key='-NUM1-'), sg.Input(key='-NUM2-')], # Pola wejściowe
    [sg.Button("Dodaj"), sg.Button("Odejmij")], # Przyciski do operacji
    [sg.Text("Wynik: ")], sg.Text(size=(15, 1), key='-OUTPUT-')] # Etykieta na wynik
]

# Tworzenie okna aplikacji
window = sg.Window("Kalkulator", layout)
```

## PRZYKŁAD 1 - KALKULATOR

```
# Pętla obsługująca interakcje użytkownika
while True:
    event, values = window.read()      # Odczyt zdarzeń i wartości z GUI
    if event == sg.WINDOW_CLOSED:      # Jeśli użytkownik zamknie okno
        break
    try:
        # Przetwarzanie danych wprowadzonych przez użytkownika
        num1 = float(values['-NUM1-'])   # Pobranie pierwszej liczby
        num2 = float(values['-NUM2-'])   # Pobranie drugiej liczby
```

## PRZYKŁAD 1 - KALKULATOR

```
# Obsługa kliknięć przycisków
if event == "Dodaj":
    result = num1 + num2
elif event == "Odejmij":
    result = num1 - num2

# Wyświetlenie wyniku w GUI
window['-OUTPUT-'].update(result)
except ValueError:
    # Obsługa błędu, gdy użytkownik podał niewłaściwe dane
    window['-OUTPUT-'].update("Błąd: Podaj liczby")

# Zamykanie okna po zakończeniu
window.close()
```

## PRZYKŁAD 1 - KALKULATOR

- ▶ Nowością dla Was jest try i except

try:

```
with open("plik.txt", "r") as f:
```

```
    print(f.read())
```

```
except FileNotFoundError:
```

```
    print("Plik nie istnieje!")
```

WPROWADZENIE BYŁO, DZISIAJ WIĘCEJ PRZYKŁADÓW

---

## PRZYKŁAD 2 - PRZEGŁĄDARKA PLIKÓW

```
import PySimpleGUI as sg
import os

# Layout GUI
layout = [
    [sg.Text("Wybierz katalog:"), sg.Input(key="-FOLDER-", enable_events=True), sg.FolderBrowse()],
    [sg.Listbox(values=[], size=(50, 10), key="-FILE LIST-", enable_events=True)],
    [sg.Text("Podgląd zawartości pliku:"), sg.Text(size=(50, 1), key="-FILE NAME-")],
    [sg.Multiline(size=(50, 15), key="-CONTENT-", disabled=True)],
]
```

```
# Tworzenie okna
window = sg.Window("Przeglądarka plików", layout)

while True:
    event, values = window.read()
    if event == sg.WINDOW_CLOSED:
        break
    if event == "-FOLDER-":
        # Obsługa wyboru katalogu
        folder = values["-FOLDER-"]
        try:
            file_list = os.listdir(folder) # Pobranie listy plików
            window["-FILE LIST-"].update(file_list)
        except:
            window["-FILE LIST-"].update([])
    elif event == "-FILE LIST-":
        # Obsługa wyboru pliku
        try:
            selected_file = values["-FILE LIST-"][0]
            full_path = os.path.join(values["-FOLDER-"], selected_file)
            with open(full_path, "r") as f:
                window["-CONTENT-"].update(f.read()) # Wyświetlenie zawartości pliku
            window["-FILE NAME-"].update(selected_file)
        except:
            window["-CONTENT-"].update("")
            window["-FILE NAME-"].update("")
window.close()
```

# PRZYKŁAD 2 - PRZEGŁĄDARKA PLIKÓW

```
# Tworzenie okna
window = sg.Window("Przeglądarka plików", layout)

while True:
    event, values = window.read()
    if event == sg.WINDOW_CLOSED:
        break
    if event == "-FOLDER-":
        # Obsługa wyboru katalogu
        folder = values["-FOLDER-"]
        try:
            file_list = os.listdir(folder) # Pobranie listy plików
            window["-FILE LIST-"].update(file_list)
        except:
            window["-FILE LIST-"].update([])
    elif event == "-FILE LIST-":
        # Obsługa wyboru pliku
        try:
            selected_file = values["-FILE LIST-"][0]
            full_path = os.path.join(values["-FOLDER-"], selected_file)
            with open(full_path, "r") as f:
                window["-CONTENT-"].update(f.read()) # Wyświetlenie zawartości pliku
            window["-FILE NAME-"].update(selected_file)
        except:
            window["-CONTENT-"].update("")
            window["-FILE NAME-"].update("")
window.close()
```

## 1. Lista plików:

- **os.listdir(folder)** zwraca listę plików w katalogu.
- **window["-FILE LIST-"].update(file\_list)** aktualizuje widok listy.

## 2. Podgląd zawartości:

- **with open(full\_path, "r")** otwiera plik do odczytu.
- Zawartość wyświetlana w polu Multiline.

# PRZYKŁAD TWORZENIE WYKRESU

---

```
1 import PySimpleGUI as sg
2 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
3 import matplotlib.pyplot as plt
4
5 # Funkcja do wyświetlania wykresu
6 def draw_figure(canvas, figure):
7     figure_canvas_agg = FigureCanvasTkAgg(figure, canvas)
8     figure_canvas_agg.draw()
9     figure_canvas_agg.get_tk_widget().pack(side="top", fill="both", expand=1)
10
```

# PRZYKŁAD TWORZENIE WYKRESU

---

```
# Layout GUI
layout = [
    [sg.Text("Podaj wartości (oddzielone przecinkami):"), sg.Input(key="-VALUES-")],
    [sg.Button("Rysuj wykres")],|
    [sg.Canvas(key="-CANVAS-")],
]

# Tworzenie okna
window = sg.Window("Rysowanie wykresu", layout, finalize=True)
```

# PRZYKŁAD TWORZENIE WYKRESU

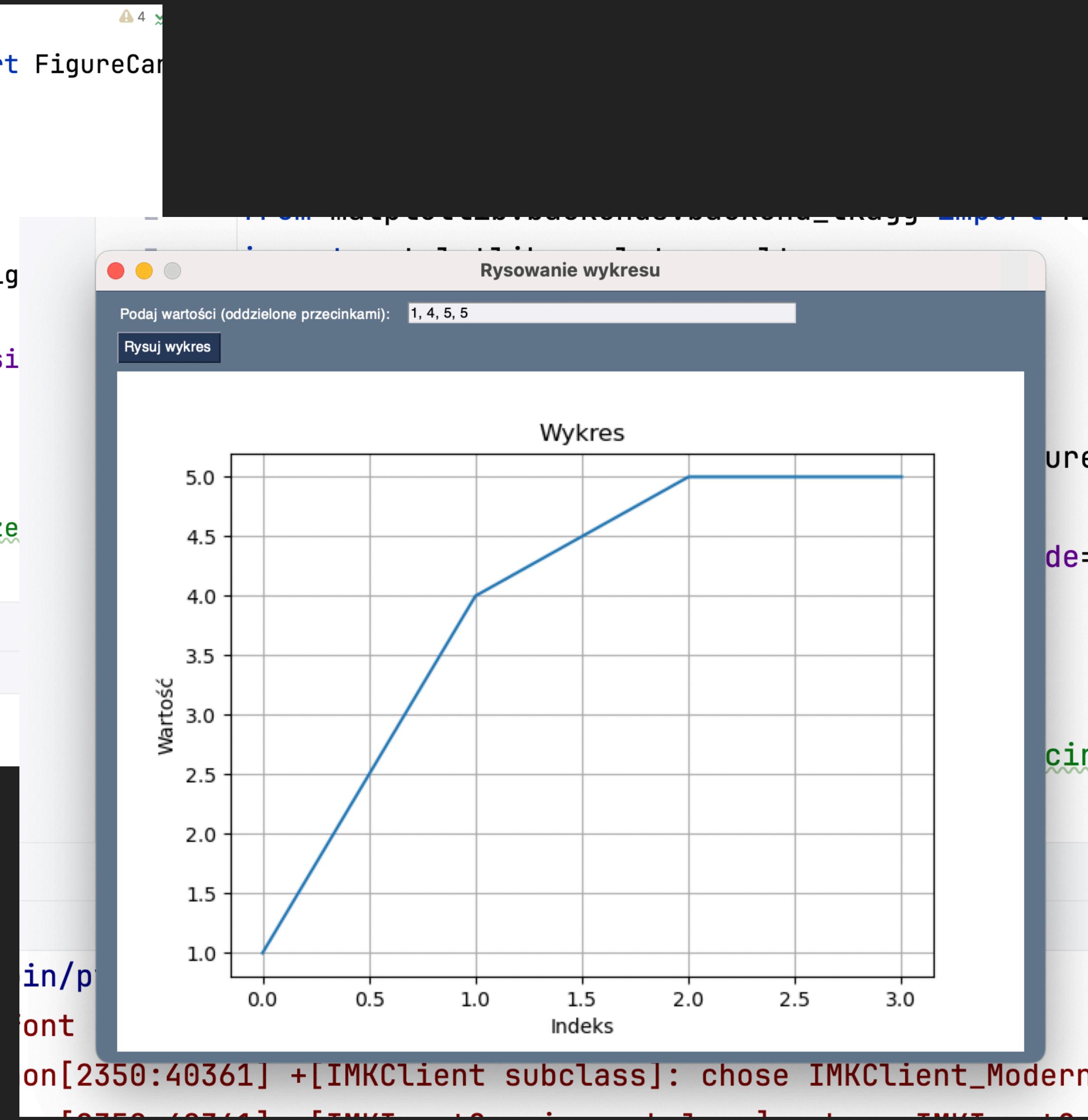
---

```
21 while True:  
22     event, values = window.read()≠  
23     if event == sg.WINDOW_CLOSED:  
24         break  
25     if event == "Rysuj wykres":  
26         try:  
27             # Odczyt danych i rysowanie wykresu  
28             data = list(map(float, values["-VALUES-"].split(",") )) # Zamiana na listę liczb  
29             plt.figure()  
30             plt.plot(data)  
31             plt.title("Wykres")  
32             plt.grid()  
33             plt.xlabel("Indeks")  
34             plt.ylabel("Wartość")  
35             draw_figure(window["-CANVAS-"].TKCanvas, plt.gcf()) # Wyświetlenie wykresu  
36         except ValueError:  
37             sg.popup("Nieprawidłowe dane")  
38 window.close()
```

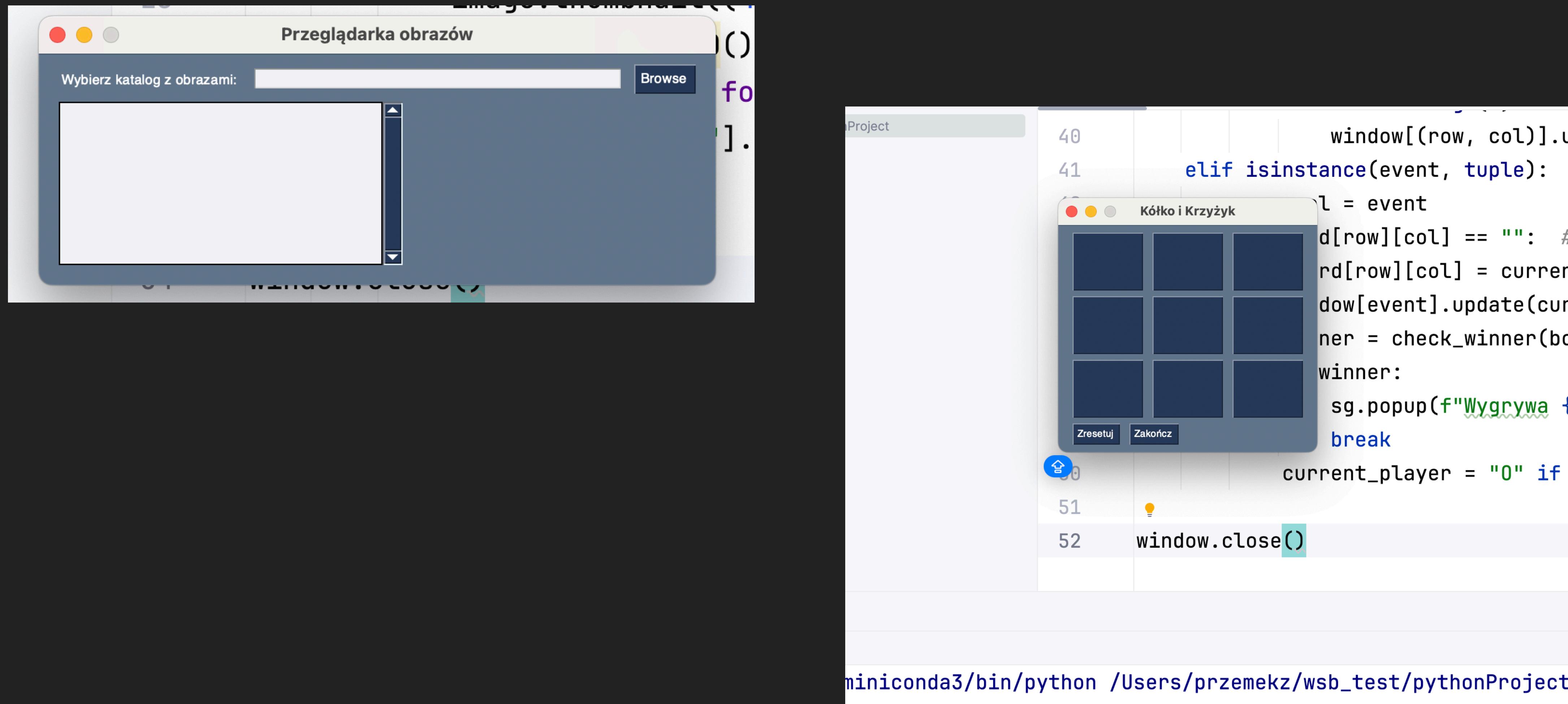
# ODSKOCZNIA - DZIAŁAJĄCE:

```
test/pythonProject
1 import PySimpleGUI as sg
2 from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
3 import matplotlib.pyplot as plt
4
5 1 usage
6 def draw_figure(canvas, figure):
7     figure_canvas_agg = FigureCanvasTkAgg(figure, canvas)
8     figure_canvas_agg.draw()
9     figure_canvas_agg.get_tk_widget().pack(side='top', fill='both', expand=1)
10
11 layout = [
12     [sg.Text("Podaj wartości (oddzielone przecinkami):")],
13     [sg.Input()],
14     [sg.Button("Rysuj wykres")],
15 ]
16
17 window = sg.Window('Rysowanie wykresu', layout)
```

```
mekz/miniconda3/bin/python /Users/przemekz/wsb_test/pythonProject/main.py
is building the font cache: this may take a moment.
```



# ODSKOCZNIA - DZIAŁAJĄCE:



```
1 import PySimpleGUI as sg
2 import random
3
4 # Ustawienia gry
5 GRID_SIZE = 20 # Wielkość jednego "kwadratu" w kratownicy
6 GRID_COUNT = 20 # Liczba kwadratów na boku planszy
7
8 # Funkcja do rysowania planszy
9 def draw_grid(window, snake, food):
10     # Wyczyszczenie planszy
11     for row in range(GRID_COUNT):
12         for col in range(GRID_COUNT):
13             color = "white"
14             if (col, row) in snake:
15                 color = "green" # Ciało węża
16             elif (col, row) == food:
17                 color = "red" # Jedzenie
18             window[(col, row)].update(background_color=color)
19
20 # Tworzenie layoutu
```

# PRZYKŁAD SNAKE

---

```
20 # Tworzenie layoutu
21 layout = [
22     [
23         sg.Graph(
24             canvas_size=(GRID_SIZE * GRID_COUNT, GRID_SIZE * GRID_COUNT),
25             graph_bottom_left=(0, 0),
26             graph_top_right=(GRID_SIZE * GRID_COUNT, GRID_SIZE * GRID_COUNT),
27             key="-GRAPH-",
28             enable_events=True,
29         )
30     ],
31     [sg.Button("Start"), sg.Button("Zakończ")]
32 ]
33
```

# PRZYKŁAD SNAKE

---

```
34 # Tworzenie okna
35 window = sg.Window("Gra w Węża", layout, finalize=True)
36
37 # Inicjalizacja planszy
38 for x in range(GRID_COUNT):
39     for y in range(GRID_COUNT):
40         window[(x, y)] = sg.Text(
41             " ", size=(2, 1), pad=(0, 0), background_color="white", key=(x, y)
42     )
43
44 # Parametry gry
45 snake = [(5, 5)]
46 direction = (0, 1) # Kierunek ruchu: w prawo
47 food = (10, 10)
48 running = False
49
```

# PRZYKŁAD SNAKE

```
50 while True:
51     event, _ = window.read(timeout=200 if running else None)
52
53     if event in (sg.WINDOW_CLOSED, "Zakończ"):
54         break
55     elif event == "Start":
56         running = True
57     elif running:
58         # Zmiana pozycji węża
59         head = snake[-1]
60         new_head = (head[0] + direction[0], head[1] + direction[1])
61
62         # Sprawdzanie kolizji
63         if (
64             new_head[0] < 0 or new_head[0] >= GRID_COUNT
65             or new_head[1] < 0 or new_head[1] >= GRID_COUNT
66             or new_head in snake
67         ):
68             sg.popup("Koniec gry!")
69             break
```

# PRZYKŁAD SNAKE

---

```
69         break
70
71     # Dodanie nowej głowy węża
72     snake.append(new_head)
73
74     # Sprawdzanie, czy wąż zjadł jedzenie
75     if new_head == food:
76         food = (random.randint(0, GRID_COUNT - 1), random.randint(0, GRID_COUNT - 1))
77     else:
78         snake.pop(0)    # Usunięcie ogona węża
79
80     # Rysowanie planszy
81     draw_grid(window, snake, food)
82
83 window.close()X
```

# PRZYKŁAD KÓŁKO I KRZYŻYK

```
1 import PySimpleGUI as sg
2
3 # Funkcja sprawdzająca zwycięstwo
4 def check_winner(board):
5     # Sprawdzenie rzędów, kolumn i przekątnych
6     for row in board:
7         if row[0] == row[1] == row[2] != "":
8             return row[0]
9     for col in range(3):
10        if board[0][col] == board[1][col] == board[2][col] != "":
11            return board[0][col]
12        if board[0][0] == board[1][1] == board[2][2] != "":
13            return board[0][0]
14        if board[0][2] == board[1][1] == board[2][0] != "":
15            return board[0][2]
16    return None
17
```

# PRZYKŁAD KÓŁKO I KRZYŻYK

---

```
18 # Tworzenie planszy
19 layout = [[sg.Button("", size=(4, 2), key=(row, col)) for col in range(3)] for row in range(3)]
20 layout.append([sg.Button("Zresetuj"), sg.Button("Zakończ")])
21
22 # Tworzenie okna
23 window = sg.Window("Kółko i Krzyżyk", layout)
24
25 # Parametry gry
26 board = [["" for _ in range(3)] for _ in range(3)] # Plansza gry
27 current_player = "X" # Zaczyna gracz X
28
```

# PRZYKŁAD KÓŁKO I KRZYŻYK

```
29 while True:
30     event, _ = window.read()
31
32     if event == sg.WINDOW_CLOSED or event == "Zakończ":
33         break
34     if event == "Zresetuj":
35         # Reset planszy
36         board = [["" for _ in range(3)] for _ in range(3)]
37         current_player = "X"
38         for row in range(3):
39             for col in range(3):
40                 window[(row, col)].update("")
41     elif isinstance(event, tuple): # Kliknięcie pola
42         row, col = event
43         if board[row][col] == "": # Sprawdzenie, czy pole jest puste
44             board[row][col] = current_player
45             window[event].update(current_player)
46             winner = check_winner(board)
47             if winner:
48                 sg.popup(f"Wygrywa {winner}!")
49                 break
50             current_player = "O" if current_player == "X" else "X" # Zmiana gracza
51
52 window.close()
```

## SKRYPTY MOŻEMY URUCHAMIAĆ BEZPOŚREDNIO W TERMINALU

- ▶ W przyszłym semestrze macie systemy operacyjne. Skrypty będą wykorzystywane np w Linuksie
- ▶ Można je dodać w czymś na kształt autostartu, czy wreszcie do usługi odpowiadającej za

WPROWADZENIE BYŁO, DZISIAJ WIĘCEJ PRZYKŁADÓW

## KOPIOWANIE I PORZĄDKOWANIE PLIKÓW

```
1 import os
2 import shutil
3
4 # Źródłowy katalog, z którego chcemy kopiować pliki
5 source_dir = os.path.expanduser("~/Dokumenty")
6
7 # Docelowy katalog, do którego pliki mają zostać skopiowane
8 destination_dir = os.path.expanduser("~/Kopia")
9
10 # Tworzenie katalogu docelowego, jeśli nie istnieje
11 os.makedirs(destination_dir, exist_ok=True)
12
13 # Iteracja po plikach w katalogu źródłowym
14 for filename in os.listdir(source_dir):
15     if filename.endswith(".txt"): # Kopiujemy tylko pliki z rozszerzeniem .txt
16         full_path = os.path.join(source_dir, filename)
17         shutil.copy(full_path, destination_dir) # Kopiowanie pliku
18         print(f"Skopiowano: {filename}")
19
20 print("Kopiowanie zakończone!")
```

# MONITOROWANIE ZUŻYCIA CPU

```
1 import psutil
2 import time
3
4 # Ścieżka do pliku, w którym zapiszemy dane
5 output_file = os.path.expanduser("~/cpu_usage.txt")
6
7 # Czas pomiędzy kolejnymi pomiarami (w sekundach)
8 interval = 5
9
10 # Otwieranie pliku w trybie do dopisywania
11 with open(output_file, "a") as file:
12     file.write("Monitorowanie zużycia CPU:\n")
13     while True:
14         # Pobieranie aktualnego użycia CPU
15         cpu_usage = psutil.cpu_percent(interval=1)
16         timestamp = time.strftime("%Y-%m-%d %H:%M:%S")
17         log_entry = f"{timestamp} - Zużycie CPU: {cpu_usage}%\n"
18
19         # Zapis do pliku
20         file.write(log_entry)
21         print(log_entry.strip())
22
23         # Przerwa między pomiarami
24         time.sleep(interval)
```

# ROBIENIE BACKUPÓW

---

```
1 import os
2 import shutil
3
4 # Katalog do backupu
5 source_dir = os.path.expanduser("~/Projekty")
6
7 # Lokalizacja pliku backupu
8 backup_file = os.path.expanduser("~/Projekty_backup.zip")
9
10 # Tworzenie archiwum ZIP
11 shutil.make_archive(base_name=backup_file.replace(".zip", ""), format="zip", root_dir=source_dir)
12
13 print(f"Backup zakończony! Plik zapisano jako: {backup_file}")
```

# KOPIOWANIE Z FTP

```
1 from ftplib import FTP
2
3 # Dane serwera FTP
4 ftp_host = "ftp.domena.com"
5 ftp_user = " użytkownik"
6 ftp_password = "hasło"
7
8 # Lokalny katalog docelowy
9 destination_dir = os.path.expanduser("~/PobraneFTP")
10
11 # Połączenie z serwerem FTP
12 ftp = FTP(ftp_host)
13 ftp.login(user=ftp_user, passwd=ftp_password)
14
15 # Pobieranie plików z katalogu głównego serwera FTP
16 os.makedirs(destination_dir, exist_ok=True)
17 for filename in ftp.nlst(): # Lista plików w bieżącym katalogu
18     with open(os.path.join(destination_dir, filename), "wb") as file:
19         ftp.retrbinary(f"RETR {filename}", file.write) # Pobieranie pliku
20         print(f"Pobrano: {filename}")
21
22 ftp.quit()
23 print("Pobieranie zakończone!")
```